COBB

INSIDE DOS

Tips & techniques for MS-DOS & PC-DOS

January 1993
Vol. 4 No. 1

Running a program at a specific time

By Van Wolverton

A number of readers have asked us to publish a technique for running a batch file at a particular time. In response to these requests, Van Wolverton has developed the SCHED.BAT file, which he outlines in this month's column.

Have you ever wanted to set a program to run later—when you won't be sitting in front of your PC? Some programs have this option built in. For example, most FAX programs include an option that lets you schedule FAX transmissions for late at night when telephone rates are lowest. You specify the name of the file that contains the FAX and the telephone number of the recipient, then leave your system running and head for bed, safe in the knowledge that your FAX will be sent while you sleep.

Letting your computer run a program at a specific time is convenient for more than just sending FAXes. You can run a communications program at night, since long-distance rates are a *lot* lower then. Or, you might have an all-afternoon meeting scheduled, but want to download your MCI Mail by 5 PM or schedule a long-running backup program to start at 2 PM. If you think this sounds useful, read on. As you'll see, you don't have to buy a special program. You can use DOS and QBasic—the programming tool Microsoft includes with DOS 5—to run a batch file or program at the time you choose. In this article, we'll show you how to set up a DOS batch file that creates and uses a QBasic program as a timer. But first, let's look at why you need to call on QBasic for help.

The tool: QBasic

Your computer uses a battery-operated clock to keep track of the time and date, even when it's turned off. Each time you start the system, DOS checks the battery-operated clock and sets its own clock, which it uses to mark the directory entry of each file with the time and date it was created or last changed.

DOS doesn't let you do much with the time and date except change them using the DATE and TIME commands or display them using the PROMPT command. But the QBasic language interpreter that comes with DOS lets you use the time and date to control what a program does. As we'll show you in a moment, combining a simple QBasic program with a batch file lets you run a program at any time you like, whether or not you're there.

The relationship between DOS and Basic programming goes way back. Since version 1, DOS has included a program that lets you write programs in some variety of the Basic programming language. MS-DOS versions through 4.01 provided a program called GW-Basic, while PC-DOS provided Basic or Basica. Starting with version 5, DOS includes a program called QBasic which, although it looks different from the others (it uses a mouse and menus), still performs the same essential function of letting you write and run Basic language programs.

The technique

You don't need to know anything about QBasic to use this technique of running a program at a specified time. To simplify things as much as possible, we'll use a single batch file to accomplish everything—create the QBasic timer and run the program at the proper time. (That batch file, called SCHED.BAT, is shown in Figure A on the next page.) After your program has finished running, the batch file erases the QBasic program and returns control to DOS.

IN THIS ISSUE

Taking advantage of parameters

To use this technique, you must provide SCHED.BAT with at least two parameters: the time to launch the program and the command required to run it. (As you may recall, parameters are the words or numbers you type after the name of the batch file when you run it.) SCHED.BAT then uses the information you provide in the parameters to create a customized QBasic timer program called TIMER.BAS. The trick is that later—when it's time to run the program—TIMER.BAS uses SCHED.BAT to start the program. But TIMER.BAS uses the word *run* as the first parameter, and the command required to run the application program as the second parameter. Basically, SCHED.BAT decides what it should do by checking on the first parameter, %1, entered with the command name:

- If the first parameter is not the word *run*, SCHED.BAT assumes that the first parameter is the time at which you want to run the application program. It then creates a QBasic program named TIMER.BAS that checks the time and, at the appointed hour and minute, runs SCHED.BAT again, specifying *run* as the first parameter.
- If the first parameter is *run*, SCHED.BAT knows that TIMER.BAS is using it to run your application program.

Writing a batch file to perform double duty like this makes understanding how the batch file works a bit more complicated. However, this method reduces the number of batch files you have to write and keep track of.

Figure A

```
@echo off
if %1==run goto RUN_IT
echo cls > timer.bas
echo while not left$(time$,5) = "%1" >> timer.bas
echo    locate 10,20  >> timer.bas
echo    print "Waiting to run %2 at %1" >> timer.bas
echo    locate 12,30 >> timer.bas
echo    print time$ >> timer.bas
echo    locate 14,22 >> timer.bas
echo    print "Press any key to cancel." >> timer.bas
echo    if not len(inkey$) = 0 then system >> timer.bas
echo wend >> timer.bas
echo shell "sched run %2 %3 %4" >> timer.bas
echo system >> timer.bas
qbasic /run timer
del timer.bas
goto END
:RUN_IT
call %2 %3 %4
:END
```

SCHED.BAT works in tandem with a QBasic program to run a program at the time you specify.

INSIDE DOS™

Inside DOS (ISSN 1049-5320) is published monthly by The Cobb Group.

Prices
Domestic ..... $49/yr. ($6.00 each)
Outside U.S. ..... $69/yr. ($8.50 each)

Phone
Toll free ..... (800) 223-8720
Local ..... (502) 491-1900
Customer Relations FAX ..... (502) 491-8050
Editorial Department FAX ..... (502) 491-4200

Address
You may address tips, special requests, and other correspondence to:

The Editor, *Inside DOS*
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

For subscriptions, fulfillment questions, and requests for bulk orders, address your letters to:

Customer Relations
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

Postmaster
Second class postage is paid in Louisville, KY. Send address changes to:

Inside DOS
P.O. Box 35160
Louisville, KY 40232

Back Issues
To order back issues, call Customer Relations at (800) 223-8720. Back issues cost $6.00 each, $8.50 outside the U.S., and you can pay with MasterCard, VISA, Discover, or American Express, or we can bill you. Please identify the issue you want by the month and year it was published. Customer Relations can also provide you with an issue-by-issue listing of all the articles that have appeared in *Inside DOS*.

Staff
Editor-in-Chief ..... Suzanne Thornberry
Contributing Editors ..... Van Wolverton
David Reid
Editing ..... Linda Watkins
Polly Blakemore
Production Artist ..... Margueriete Stith
Design ..... Karl Feige
Publications Manager ..... Elayne Noltemeyer
Managing Editor ..... Jim Welp
Circulation Manager ..... Brent Shean
Publications Director ..... Linda Baughman
Editorial Director ..... Jeff Yocom
Publisher ..... Douglas Cobb

Advisory Board
Earl Berry Jr.
Tina Covington
Marvin D. Livingood

Creating a file with a batch file

So far, we've focused on how SCHED.BAT uses parameters to decide what to do. We've also mentioned that SCHED.BAT uses the time you enter as the first parameter to create a customized timer program named TIMER.BAS. You might be wondering how a batch file can create a QBasic program. Actually, the technique s simple.

You can create a file (even a program file) by redirecting the output of the ECHO command to a file. For example, if you type the following command:

```
C:\>echo This is FRED > fred
```

DOS will create a file named FRED that consists of one line:

```
This is FRED
```

You can add lines to the file using the >> redirection symbol. Typing the following commands, for example, will add two more lines to FRED:

```
C:\>echo This is the second line >> fred
C:\>echo And this is the last line >> fred
```

Of course, the FRED file created by these statements is a plain text file. Fortunately, a plain text file can just as easily contain QBasic commands as declarations about Fred. As you can see in Figure A, SCHED.BAT uses 12 ECHO statements to create a TIMER.BAS program, line by line. In effect, each time you run SCHED.BAT, the batch file totally re-creates the QBasic program. This might seem inefficient at first; after all, why not just create a single TIMER.BAS program and run it directly from the batch file?

The answer lies in the parameters used in several of the ECHO statements. By creating the TIMER.BAS file each time you schedule a program, SCHED.BAT can customize TIMER.BAS. You enter the parameters for the time you want to run the program and the program name, and SCHED.BAT will echo the information into the TIMER.BAS program.

SCHED.BAT: a dual-purpose batch file

Now that we've given you some background on how SCHED.BAT creates and uses a QBasic program, let's look more closely at what each section of SCHED.BAT does. As with any batch file, you can create SCHED.BAT using the DOS 5 Editor or another word processor that allows you to create ASCII-only text files.

The first line is the usual ECHO command, which eliminates screen clutter. The remainder of SCHED.BAT is made up of four functional parts:

1. The IF command in line 2 checks to see whether the first parameter is the word *run*; if it is, DOS skips to the :RUN_IT label. As you'll see, this is how SCHED.BAT tells whether you or the QBasic program is using the SCHED command.

2. The ECHO commands in lines 3 through 14 create the QBasic program in a file named TIMER.BAS. SCHED.BAT executes these commands if you use the SCHED command, but not if the QBasic program uses it. Notice that we create the file by using the > redirection symbol in line 3; the other redirection symbols are >>, which add statements to an existing file.

3. Lines 16 through 18 tell QBasic to run the program named TIMER (you don't have to specify the extension BAS), delete the program after running it, and then go to the end of the batch file (returning to DOS).

4. The :RUN_IT label in line 19 and the CALL command in line 20 actually run the program you specified. TIMER.BAS executes the CALL command when it uses the SCHED command and specifies *run* as the first parameter instead of a time. The %2 parameter is the program name you enter when running SCHED.BAT. If needed, the %3 and %4 parameters in this CALL command will represent the first two parameters of the program you want to run (%2). You can even add a %5 parameter if the program you run needs three parameters.

TIMER.BAS: the disappearing QBasic program

As we've shown, the ECHO commands in lines 3 through 8 of SCHED.BAT use replaceable parameters (%1 and %2) to create the TIMER.BAS program, which runs the program represented by %2 at the time represented by %1. Now, let's look more closely at this elusive QBasic program. Suppose you want to run at midnight a batch file named DISKBAK.BAT, which will back up your entire hard disk to a tape device. You simply type *sched 24:00 diskbak* and press [Enter]. Then, SCHED.BAT will create and run the QBasic program shown in Figure B on the next page.

Figure B

```
cls
while not left$(time$,5) = "24:00"
locate 10,20
print "Waiting to run diskbak at 24:00"
locate 12,30
print time$
locate 14,22
print "Press any key to cancel."
if not len(inkey$) = 0 then system
wend
shell "sched run DISKBAK"
system
```

This temporary TIMER.BAS program runs the DISKBAK.BAT batch file at midnight.

Even if you're not familiar with the QBasic language, the program is pretty straightforward:

- The first statement (CLS) clears the screen, just like the DOS CLS command.
- The group of statements starting with line 2 (WHILE . . .) and ending with line 10 (WEND) execute repeatedly as long as the current time (TIME$) isn't equal to the time you specified in the SCHED command ("24:00").
- The series of PRINT and LOCATE statements displays the message *Waiting to run diskbak at 24:00* starting in row 10, column 20; the current time starting in row 12, column 30; and *Press any key to cancel* starting in row 14, column 22.
- The IF statement in line 9 checks to see if you have pressed a key. If you have, TIMER.BAS ends QBasic and returns to DOS without running the program you specified.
- TIMER.BAS executes the SHELL statement in line 11 only when the current time is equal to the time you specified in the SCHED command. This SHELL statement tells DOS to carry out the SCHED command with the RUN DISKBAK parameters, which run the program you specified.
- After your program ends, the SYSTEM statement in line 12 ends QBasic, which returns control to SCHED.BAT. The remaining commands in SCHED.BAT delete TIMER.BAS (you don't need it any more) and return to the DOS command line.

(Of course, each TIMER.BAS file will vary slightly to refer to the time and program you specify.)

Scheduling your jobs

Once you've created SCHED.BAT, you can schedule programs to run at a future time, using the syntax

```
C:\>sched time program_name [parameter1] [parameter2]
```

To satisfy the way QBasic looks at the time, you must enter 0 before single digit hours and minutes (5 minutes past 1 is 01:05 in the morning, but it's 13:05 in the afternoon). Include a drive letter and path if the program you specify isn't in the current directory or a directory in your command path. The last two parameters, which are shown in brackets, are optional. You can use them if you need to pass one or two parameters to the program you want SCHED.BAT to run for you.

For example, suppose you want to run Procomm Plus (the PCPLUS.EXE file) at 4 AM to transfer some files. Let's also suppose that the commands to dial, transfer the files, and end Procomm Plus are in a script file named FTRANS.ASX. To run this program, you enter:

```
C:\PROCOMM>sched 04:00 pcplus /ftrans.asx
```

If you run Procomm Plus every night, you can store the command in a batch file (you could call it GNITE.BAT).

If you have programs set up to return to DOS automatically—as does our example script Procomm FTRANS.ASX—you can run several programs in a sequence. Just use the CALL command to ensure that the programs return to SCHED.BAT when they quit running, and remember to leave sufficient running time. For example, suppose you want to run DISKBAK, Procomm Plus, and a disk defragmentation program called DEFRAG; the first runs for about 20 minutes, and the second for 10 minutes. You can put the following commands in the batch file named GNITE.BAT:

```
call sched 02:00 diskbak
call sched 02:30 pcplus /ftrans.asx
call sched 02:45 defrag
```

Now, just type *gnite* and head for home (or bed). It will all be done tomorrow morning.

Note

Since QBasic uses about 277 Kb of RAM as it runs TIMER.BAS, you may receive *Insufficient memory* errors if you try to schedule large programs. However, we've been able to successfully run programs that use less than 300 Kb of RAM.

Contributing editor Van Wolverton is the author of the best-selling books Running MS-DOS 5 *and* Supercharging MS-DOS. *Van currently lives in Alberton, Montana.*

Getting into the BACKUP habit

Computer sages—including *Inside DOS* columnist Van Wolverton—often advise us to back up our hard disk regularly. Few of us, however, live up to this ideal. After all, backing up your hard disk always seems less important than finishing that special project or starting the weekend. Your computer's reliability probably encourages a lax attitude: It always boots up, and your data is always there.

In reality, information stored on a magnetic disk is inherently unstable. A magnetic field can make your data vanish. Electrical surges caused by lightning or flaky power supplies can zap data on your hard disk. Mechanical failure can cause the read-write arm of your hard disk to crash into the platter that stores data—the origin of the phrase "My system crashed!" With electromagnetic and mechanical failure lurking as potential enemies, some experts say that the question isn't *if* you'll lose data on your hard disk, but *when* will it happen.

When you're developing a backup strategy, decide exactly what you need to back up and how often you should back up.

In this article, we'll show you the basics of using the BACKUP command. We'll also show you how being selective about what to back up can make your routine backups go more quickly. But first, let's review some basics about the BACKUP command.

Backup basics

The BACKUP command that DOS supplies copies files in a special format. You can read these files only by restoring them with—you guessed it—the RESTORE command. (We explain more about using that command in the article "The RESTORE Command Works Best When You're Specific," which begins on page 9.) The BACKUP command doesn't noticeably compress files. However, it does allow DOS to pack diskettes full because it is able to split files between diskettes. (We used this BACKUP capability in the article "The BACKUP Command Lets You Copy a Large File onto Multiple Diskettes," which appeared in the October 1992 issue of *Inside DOS*.)

Keep in mind...

Before you back up your hard disk, you should be aware of a few points about the BACKUP and RESTORE commands. A very important consideration is that you can't use BACKUP or RESTORE when you're using the ASSIGN, JOIN, or SUBST commands. These commands alias drives or directories, which can confuse the BACKUP and RESTORE commands.

Also, if you have your hard disk partitioned into several drives, remember that you must back up each partition. These partitions are also known as *logical drives*. For example, if you have three partitions on your hard disk, you'll need to back up the C:, D:, and E: drives.

You should also be careful of backing up copy-protected files. Many software companies copy-protect their program files to shield them from software pirates. If you try to back up a program containing copy-protected files, you may not be able to restore the program to your hard disk. As long as you have the original program diskettes, the solution is simple: Just re-install the program.

Generally, DOS' BACKUP and RESTORE commands require more time and effort than third-party backup utilities. Depending on the amount of data on your hard disk drive and the speed of your disk drives, you may spend 20, 30, or even 40 minutes backing up a hard disk. If you let BACKUP format diskettes as it backs up your hard disk, you may nearly double the time required to back up your system. (In the article "Letting BACKUP Format Diskettes" on page 9, we explain some special considerations about formatting as you back up.)

Backup strategies

Many people automatically think of backing up the entire hard disk whenever they think of the BACKUP command. But DOS actually gives you several intermediate options. In this article, we'll focus on three common BACKUP techniques:

- Backing up your entire hard disk
- Backing up files changed since the last backup
- Backing up data directories

When you're developing a backup strategy, decide exactly what you need to back up and how often you should back up. Let's look more closely at these decisions.

First, let's consider *what* you should back up. Do you really need to back up your programs? Probably not. After

all, you should have your original installation diskettes, as well as the copies of the diskettes that most installation guides suggest you create before installing a new program on your PC.

If you keep your data files in a directory separate from your program files, you can simply back up your data directories. We explained the importance of keeping data files separate from program files in "Strategies for Organizing Directories," which appeared in the August 1992 issue of *Inside DOS*. But, if you mix data files and program files, you have three choices: move the files to a data directory; specify the names of individual files with the BACKUP command; or back up the entire program directory. Of course, if you back up your entire hard disk, you'll capture all of your data files, no matter where you've stored them. Let's take a closer look at this widely used form of the BACKUP command.

Backing up your hard disk

At some point, you'll probably want to back up your entire hard disk. In fact, some hard disk utilities recommend that you back up your entire hard disk before performing certain procedures, such as defragmenting files. You can back up an entire hard disk drive by typing *backup*, a space, the letter of the drive you want to back up, a colon, a backslash, the *.* wildcard file specification, a space, the letter of the drive to receive the backup, another space, and then the /S switch. The /S switch tells DOS to record the directory structure of the disk and back up the files in the subdirectories.

For example, suppose you want to back up your C: drive to diskettes in your A: drive. To do this, you enter the following BACKUP command:

```
C:\>backup c:\*.* a: /s
```

The BACKUP command then will present the message shown below, telling you to insert the first backup diskette and press a key to continue:

```
Insert backup diskette 01 in drive A:

WARNING! Files in the target drive
A:\ root directory will be erased
Press any key to continue. . .
```

Once you've inserted the disk, you can press a key to continue. If you insert an unformatted diskette into the drive, DOS will display the message *Checking existing disk format* and *Formatting*, followed by the diskette capacity. Once DOS has determined that the diskette is formatted, it will begin listing the path and name of each file it backs up, as follows:

```
*** Backing up files to drive A: ***
Diskette Number: 01

\AUTOEXEC.BAT
\CONFIG.SYS
\MIRROSAV.FIL
\MIRROR.FIL
```

When the diskette is full, BACKUP will prompt you for the next one:

```
Insert backup diskette 02 in drive A:
```

followed by the familiar warning message. At this point, you should remove the first diskette and write the number 1 on its label with a felt tip marker. (Never use a ball point pen or press hard when you write on a diskette label.) Then, insert the second diskette into the drive and press a key. BACKUP will continue backing up files. Continue this process, inserting a new diskette when prompted, until the backup is completed.

Remember: If you've partitioned your hard disk into several logical drives, you'll need to back up each drive. So, if you have a D: drive, you'd issue the command

```
C:\>backup d:\*.* a: /s
```

after backing up your C: drive.

Backing up files changed since your last backup

If you have more than 10 Mb or so of data on your hard disk, you probably won't want to do a complete backup every week. Fortunately, you can tell the BACKUP command to back up only the files changed since the last backup. This process is known as an *incremental backup*.

You can perform an incremental backup because DOS turns off a file's archive attribute after you back up the file. However, when you change the file later, DOS will turn on the archive attribute. New files also have the archive attribute turned on. You can see which files you've changed since the last backup by issuing the ATTRIB command. Adding the MORE filter to the command will display the list one screen at a time. For example, to see which files you've changed in your C:\BATCH directory, change to the directory, then enter the following command:

```
C:\BATCH>attrib *.* | more
```

DOS will display the letter *A* to the left of the files you've created or updated since the last backup.

You tell DOS to perform an incremental backup by adding two more switches to the BACKUP command. The /A switch tells the BACKUP command to

append the updated files to the last diskette from your most recent backup. (Regardless of whether DOS prompts you for the last diskette, it's important to use the last diskette from your previous backup when you do an incremental backup.) Then, you add the /M switch to back up only files that have been changed since the last backup—that is, files with the archive attribute turned on.

When you perform an incremental backup, DOS will append modified files to your previous backup. If you later restore these backups, DOS will restore a file as many times as it was backed up. Each time, the RESTORE command will overwrite the file with its more recent backup. As you might imagine, the RESTORE command becomes less efficient with each incremental backup, since it ends up repeatedly overwriting modified files. For this reason, many DOS users do a full backup once a month, then three incremental backups (one per week).

If you use this method, we suggest that you keep two sets of backup diskettes. When you do your second monthly backup, use new diskettes. In the third month, you can reformat the diskettes from the first month. Alternating the sets and reformatting them guards against bad clusters in your diskettes, which could render your backups useless.

How many diskettes will you need?

When you're getting ready to back up your hard disk, you'll need to have enough diskettes to hold the complete backup. If you're backing up the entire hard disk, you can estimate how many diskettes you'll need by changing to the drive you want to back up and issuing the CHKDSK command. The safest strategy for estimating how much data you need to back up is to subtract the *bytes available on disk* from the *bytes total disk space*:

```
  bytes total disk space
- bytes available on disk
  bytes needed for backup
```

Then, you'll need to divide the bytes you estimate for your backup by the bytes available on a formatted diskette. Table A provides this information.

Table A

Diskette size	Number of bytes
360 Kb	362,496
720 Kb	730,112
1.2 Mb	1,213,952
1.44 Mb	1,457,664

You can use this table to find the number of bytes your diskettes can hold.

(These aren't convenient, round numbers because a kilobyte is 1024, rather than 1000, bytes.)

For example, suppose you want to back up your C: drive on 1.2 Mb, 5.25" diskettes. You can run CHKDSK to estimate how many diskettes you'll need for the backup. Suppose you run CHKDSK on your C: drive and come up with the following result:

```
C:\>chkdsk
Volume COBB      created Jul 14, 1992  4:33p

     33462272 bytes total disk space
        75776 bytes in 4 hidden files
        79872 bytes in 30 directories
     22751232 bytes in 879 user files
     10555392 bytes available on disk

       655360 bytes total memory
       581312 bytes free
```

Now, to determine the number of bytes you'll need to back up, subtract the bytes available on disk from the bytes total disk space, as we've done below:

```
  33462272  bytes total disk space
-10555392  bytes available on disk
  22906880  bytes to back up
```

Finally, to estimate the number of 1.2 Mb diskettes needed for the backup, divide the number of bytes to back up (22,906,880) by the capacity of the 1.2 Mb diskettes (1,213,952):

```
  22906880  bytes to back up
/1213952  bytes per diskette
     18.87  (at least 19 diskettes)
```

We've rounded the result to 18.87, which means that you'll need at least nineteen, 1.2 Mb diskettes to back up your hard disk. For safety's sake, have 20 on hand in case a diskette is unusable. If you've partitioned your hard disk into several logical drives, you'll have to repeat the CHKDSK command and calculation for each drive.

Returning to our example, suppose a week or so has passed since your last backup. To perform an incremental backup of your C: drive, you first must find the last diskette in the most recent backup. So, if your backup requires 19 diskettes, you'd insert the nineteenth diskette into the A: drive. Then, issue the following command:

```
C:\>backup c:\*.* /s /a /m
```

As you can see, you still include the /S switch to tell the BACKUP command to look for files in all the directories and subdirectories.

Backing up selected directories

As we noted earlier, backing up selected directories is a great strategy—if you're careful to keep your data files in directories separate from your program files. To back up the files in a single directory, just include the directory name in the BACKUP command. You can use the *.* wildcard file specification to back up all the files. For example, to back up all of your Word documents in the C:\DOCS directory, you'd issue the following command:

```
C:\>backup c:\docs\*.* a:
```

Of course, if you've created subdirectories within the C:\DOCS directory, you can include the /S switch:

```
C:\>backup c:\docs\*.* a: /s
```

Or, if you always use the same extension for the files, you can specify the extension instead of the second asterisk. This technique can save time and diskette space, especially if your program creates its own backup files. For example, Microsoft Word uses the DOC extension as a default for the current versions of documents, and BAK for the backup versions it creates. If you always use the default DOC extension, you can back up current files by issuing the following command:

```
C:\>backup c:\docs\*.doc a: /s
```

Chances are, you'll need to back up more than one data file. In addition to a C:\DOCS directory, for example, you might have directories named C:\SHEETS for spreadsheets and C:\DATA for database files. You can list the backup commands you need in a simple batch file. Running the batch file for backups will help ensure that you don't forget an important data directory.

Note

If you receive the message *Too many open files* when you try to use the BACKUP command, you probably need to adjust the FILES setting in your CONFIG.SYS file. Using the DOS Editor or an ASCII-compatible word processor, add the following line to your CONFIG.SYS file:

```
files=30
```

If you already have a FILES setting in your CONFIG.SYS, increase it to 30. If you already have a setting of 30 or more files, try increasing the number of files by 10.

Conclusion

In this article, we've shown you how to use the BACKUP command to safeguard your data. You can use the BACKUP command to back up your entire hard disk, but this isn't always the wisest choice. We looked at two alternatives: backing up files changed since your last complete backup, and backing up only data directories.

For more information on protecting your data, see the following article:

"Van Wolverton: Protecting Your Data and Equipment from Disaster," July 1992

See the masthead on page 2 for information on ordering back issues.

QUICK TIP

A clue about diskette density

As we mentioned in the article "Choosing the Right Format Option for the Right Diskette," which appeared in the October 1992 issue, 3.5" diskettes are imprinted with an *HD* logo to indicate that they are high density (1.44 Mb) diskettes. You can assume that 3.5" diskettes without the logo are double density (720 Kb). Unfortunately, it's a bit trickier to tell the difference between 5.25" diskettes.

If you look carefully at two 5.25" diskettes of different capacities, however, you'll probably be able to notice a subtle difference. The trick is to look at the circular cutout in the diskette's plastic covering. The 360 Kb, double-density diskette has an inner ring of a different color. Usually, this ring is white, which stands out noticeably. Be careful, though: Some manufacturers use a dark gray ring, which provides little contrast. On the other hand, the 1.2 Mb, high-density diskette has no additional ring. You simply see a bit of the thin, black film of the magnetic media beneath the protective covering.

Letting BACKUP format diskettes

DOS 5's BACKUP command will automatically format a new diskette if you place it in the target drive. Although this is a handy feature, letting the BACKUP command format diskettes may almost double the time it takes to back up your hard disk. Furthermore, you need to take a couple of precautions when you want BACKUP to format diskettes.

Use the highest density diskettes your drive accepts

If you decide to let the BACKUP command format the diskettes, remember that it will format the diskettes to the maximum capacity of your drive. Normally, this is exactly the formatting you want. After all, you would probably rather back up onto twenty 1.2 Mb, 5.25" diskettes than sixty 360 Kb, 5.25" diskettes. To ensure that your backup diskettes are readable, choose the highest capacity diskettes your drive supports. (We explain more about selecting diskettes in "Choosing the Right Format Option for the Right Diskette," which appeared in the October 1992 issue of *Inside DOS*.)

Watch out for old copies of FORMAT.COM

When you let the BACKUP command format diskettes for you, you may receive a puzzling error message during the process. After you insert an unformatted diskette into the backup drive, DOS will present its usual warning about erasing all files from the diskette's root directory. When you press a key to continue, DOS will display the following error message:

```
Incorrect DOS version
```

The BACKUP command then attempts to back up files to the unformatted diskette, resulting in the error

```
General failure reading drive A
Abort, Retry, Fail?
```

This error occurs when the BACKUP command finds an old copy of FORMAT.COM in a directory it backs up. When BACKUP next attempts to format a diskette, it tries to use the old version of FORMAT.COM instead of the DOS 5 version. While this may sound like an unlikely occurrence, many users have old versions of FORMAT.COM on their hard disks. These old versions of the command are in the OLDDOS directory the DOS 5 Setup program creates when you install DOS 5.

To avoid this situation, you can simply rename the FORMAT.COM file stored in your C:\OLDDOS directory. Just issue the following command *before* you run BACKUP:

```
C:\>ren c:\olddos\format.com c:\olddos\format.old
```

Once you've renamed the old FORMAT.COM, the BACKUP command won't try to use it to format a diskette. Of course, if you ever need to use the old version of FORMAT.COM again, you'd have to rename FORMAT.OLD to FORMAT.COM.

The RESTORE command works best when you're specific

In "Getting into the BACKUP Habit," which begins on page 5, we focus on using the BACKUP command to safeguard your data. This command creates files that you can read only by using the RESTORE command. However, unless you use the BACKUP and RESTORE commands to copy large files (as we described in "The BACKUP Command Lets You Copy a Large File onto Multiple Diskettes" in the October issue), you probably haven't had much practice using the RESTORE command. After all, you'd only need to restore the files on your hard disk if disaster struck.

In this article, we'll present two tips for using the RESTORE command successfully. We'll begin by showing you the most powerful form of the RESTORE command—the "Golden" RESTORE.

Using the "Golden" RESTORE command

When you back up your hard disk, you use the BACKUP command with the /S switch. This switch tells DOS to back up all files, beginning with those in the current directory, and to proceed through the subdirectories.

As you might expect, when you restore backups that include subdirectories, you need to include the /S switch with the RESTORE command. For example, if you backed

up your entire C: drive with the command

```
C:\>backup c:\*.* a: /s
```

you'd need to issue the following RESTORE command to restore the disk drive:

```
C:\>restore a: c:\*.* /s
```

This form of the RESTORE command—which includes the *.* wildcard file specification and the /S switch—is often called the "Golden" RESTORE command because it works almost universally. You will definitely need to use the "Golden" RESTORE command if you ever need to restore all the files to your hard disk.

Restoring a single directory

Ironically, restoring files to a directory requires a bit more thought than restoring files to your entire disk. If you've backed up a particular data directory and then need to restore it, you must specify the directory's name in the RESTORE command. For example, suppose you need to restore the spreadsheets originally stored in the C:\SHEETS directory. If your C:\SHEETS directory no longer exists, you'll need to make a new C:\SHEETS directory *before* you restore the files. To do this, simply change to the root directory on your C: drive, then issue the command

```
C:\>md sheets
```

Now, to restore all of the files and subdirectories in the C:\SHEETS directory, simply issue the following command:

```
C:\>restore a: c:\sheets\*.* /s
```

Again, we've included the *.* file specification and the /S switch to ensure that the command restores all backed up files and any subdirectories that were backed up from the C:\SHEETS directory.

Conclusion

In this article, we've shown you the basics of using the RESTORE command. The most powerful form of the command—the "Golden" RESTORE—uses the *.* wildcard file specification and the /S switch.

RESTORE WORKAROUND

Listing the files on your backup diskettes

Recently, Robert W. Houston of Gaithersburg, Maryland, asked if we knew of a way to make DOS display the files stored on a backup diskette. As Mr. Houston pointed out, issuing the RESTORE command with the /D switch is supposed to display the backup files on a diskette without actually restoring the files. Unfortunately, this feature doesn't work properly in DOS 5.0.

You can display the files on the first diskette in your backup set by inserting the diskette in the drive (we'll assume you're using the A: drive), and then entering the following command:

```
C:\>restore a: c: /d /s
```

DOS will then present the message

```
Insert backup diskette 01 in drive A:
Press any key to continue. . .
```

Since you've already inserted the diskette in the A: drive, simply press a key. DOS will respond by listing all of the files you backed up on the first diskette. At the end of the list, DOS will present the message

```
Insert backup diskette 02 in drive A:
Press any key to continue. . .
```

When you insert the second backup diskette in the A: drive, then press a key, DOS will present an ominous message:

```
Restore file sequence error
```

The workaround

Fortunately, there's a simple solution to this problem. When you see the message prompting you to insert the second disk, leave the first disk in the drive and press a key. DOS will display the message

```
WARNING! Diskette is out of sequence
Replace diskette or continue if OK
Press any key to continue. . .
```

At this point, you can place the second diskette in the A: drive and press a key. DOS will then list the files on the diskette. You'll need to repeat this technique for the whole backup set. Whenever DOS prompts you for the next diskette, simply leave the current diskette in the drive and press a key. After DOS displays the warning, insert the next diskette and press a key.

A variation on copying miscellaneous files to a diskette

William Groce of Princeton, New Jersey, submitted the COPI.BAT batch file featured in this article.

As you may know, the COPY command doesn't allow you to specify multiple files to copy unless you represent the files with a wildcard file specification. For example, you can copy all text files from the current directory to your A: drive by typing *copy *.txt a:* and pressing [Enter]. But suppose you want to copy files with different names, or perhaps even different paths? For example, suppose you want to copy to your A: drive a spreadsheet named C:\DATA\BUDGET.XLS, a mailing list named C:\DATA\MAILING.DAT, and a report named C:\DATA\WINTER.DOC. If you issue the command

```
C:\DATA>copy budget.xls mailing.dat winter.doc a:
```

DOS will present the message *Too many parameters.* However, you can add a plus sign (+) between the filenames, and DOS will copy the three files into a single file named WINTER.DOC on the A: drive. Although A:\WINTER.DOC will contain all of the data in the three source files, you probably won't be able to use the concatenated information.

Figure A

```
@echo off
rem William Groce's COPI.BAT copies the files you specify
rem to a diskette drive
if "%1"=="" goto :error
for %%a in (a A) do if "%%a"=="%1" goto :a
for %%a in (b B) do if "%%a"=="%1" goto :b
:a
shift
:newa
if "%1"=="!" goto :end
echo Copying %1
copy %1 a:
shift
goto :newa
:b
shift
:newb
if "%1"=="!" goto :end
echo Copying %1
copy %1 b:
shift
goto :newb
:error
echo To copy a series of files to the A: or B: drive, type
echo COPI drive [filename.ext] [filename.ext]. . . !
echo Replace drive with a or b to indicate the destination
echo Be sure to end the list by typing a space and !
:end
```

You can create COPI.BAT by entering these lines in the DOS Editor.

Subscriber William Groce faced this problem and developed a batch file to solve it. COPI.BAT, shown in Figure A, allows you to copy a number of files to a diskette in the A: or B: drive. (COPI.BAT is a more flexible version of Brad Hassler's COP.BAT, which appeared in the June 1991 issue of *Inside DOS*.) You run COPI.BAT by typing *copi*, a space, and *a* or *b* to indicate which diskette drive you want to receive the files. After the drive letter, type a space and the names of the files you want to copy, each separated by a space. Finally, you type a space and an exclamation point to end the list.

Returning to our example, you can use COPI.BAT to copy the files by entering the following command:

```
C:\DATA>copi a budget.xls mailing.dat winter.doc !
```

Be sure to type the ! at the end of the list of files. If you don't, COPI.BAT will loop endlessly, and you'll need to press [Ctrl][Break], then press Y to abort the batch file. In fact, you can type as many filenames as you'd like, as long as you have room for the ! before you reach the DOS command line limit of 135 characters.

How COPI.BAT works

Let's briefly look at how this useful batch file works. As usual, the first line of COPI.BAT turns echo off so DOS won't display each line of the batch file as it runs. The REM statements allow you to place a brief note in the batch file. DOS will ignore the REM lines when you run COPI.BAT.

The next three statements determine where COPI.BAT will look for further instructions. First, the IF statement checks to see if there is a %1 parameter—that is, if you typed anything after *copi* when you ran the batch file. If there is no %1 parameter, COPI.BAT will branch to the :ERROR label. As you can see, the :ERROR section simply echoes instructions for using the batch file, and then quits.

Next, a FOR statement checks to see if you specified the A: drive as the destination for the files:

```
for %%a in (a A) do if "%%a"=="%1" goto :a
```

Because you placed both *a* and *A* in the set, the FOR command will allow you to type either the uppercase or lowercase drive letter when you use COPI.BAT. In either case, the FOR command will go to the :A section of the batch file. (Similarly, the second FOR command jumps to the :B section if you've typed either *b* or *B* as the first parameter.)

Microsoft Technical Support
(206) 454-2030

Please include account number from label with any correspondence.

The :A section contains only a SHIFT command. This command tells DOS to shift the parameters you enter by one: %2 becomes %1, %3 becomes %2, and so on. Without the SHIFT command, COPI.BAT can accept only eight file specifications for copying; with SHIFT, you're limited only by the command-line length. After executing the SHIFT command, COPI.BAT carries out the instructions in the :NEWA section. The first instruction under :NEWA checks to see if the batch file has shifted the ! to the first parameter. Since ! indicates the end of the list, the batch file ends when ! is the first parameter. As long as %1 is not !, the batch file will echo a message that it's copying a file and copy the current %1 to the A: drive. After copying the file, COPI.BAT will shift the parameters again and return to the top of the :NEWA section. COPI.BAT will repeat this process until %1 is !, when it will end. As you'd expect, the :B and :NEWB sections work in the same way as the :A and :NEWA sections, except that they copy to the B: drive.

LETTERS

SUBST can prevent you from erasing directories

Your answer to "Erasing Old Programs from Your Hard Disk" in the September 1992 issue is very good and correct *if* you have hidden files in the "unremovable" directory. However, I have had directories that were absolutely empty and still couldn't remove them with the RD command. I received a *Directory not empty* message when I attempted to remove the directories.

After much head-scratching and frustration, I discovered the culprit was a long-forgotten, path-shortening SUBST command in my AUTOEXEC.BAT file. The SUBST command had the "unremovable" directory in its path and was preventing DOS from removing the directory. Once I removed the erroneous SUBST command from my AUTOEXEC.BAT file and rebooted my system, DOS was able to remove the directory without problems.

After this experience, I'd advise readers who have a troublesome directory to enter the SUBST command without parameters. SUBST will display any reassigned directories, so you'll be able to see if the troublesome directory is on the list. If it is, you can solve the problem simply by tracking down and removing the SUBST command that refers to the directory you want to remove.

Larry Christian
Orange, California

We've heard from several readers who have encountered the problem you describe, and your suggestion might help them remove the directories. Thanks for sharing your discovery with *Inside DOS*.

Displaying words with SAVER.BAS

I use the SAVER.BAS program presented in the article "Creating a Screen Saver with MS-DOS QBasic," which appeared in the September 1992 issue of *Inside DOS*. Is there a way to display a word with SAVER.BAS instead of the time?

Chris Celmer
Lincoln Park, Michigan

Yes. Just substitute the word you want to display instead of the TIME$ variable in the SAVER.BAS file. The TIME$ variable appears in the PRINT statement shown below:

```
PRINT TIME$;                ' display system time
```

To tell QBasic that you want to display exactly what you type, enclose the word or phrase in quotation marks. For example, to display *Happy New Year* on your screen, you'd change the PRINT statement in SAVER.BAS to:

```
PRINT "Happy New Year";   'display greeting
```

Be sure to place the semicolon *after* the closing quotation mark. (By the way, you don't *have* to change the comment to *display greeting*, since QBasic ignores anything preceded by a single quotation mark.)